Swami Sahajanand College of Computer Science

B.C.A. SEM-V

Subject: Software Engineering

UNIT I

Introduction, Software Requirement Analysis & Specification

- Define -Software & Software Engineering
- Software Engineering Approach – Phase Development Process, Project Management
- Software Process & It's Characteristics
- Software Development Process Models – Water Fall Model, Iterative Enhancement.
- Define Software Requirements
- Need For SRS, Role of
- Requirement Process -Problem Analysis, Requirement, Specifications, Validation

Software - Definition

- Software is a collection of computer programs that enable the user to
  interact with a computer, its hardware, or perform tasks.
- Without software, most computers would be useless.
- For example, without your Internet Browser software, you could not surf
  Internet.
- **Engineering** is the process of designing and building something that serves a particular purpose and finds a cost-effective solution to problems.
- **Software Engineering** is the process of designing, developing, testing, and maintaining software.
- The main goal of Software Engineering is to develop software applications for improving quality, budget, and time efficiency.

Applications of software:

- Softwares are use in graphics design.
- It is used in communications.
- It is broadly used in manufacturing plants. It is used in automation.
- Software is used in business.
- It is used in research work.
- It is used in medical field.
- Softwares are used in home.
- Softwares are used in transportation.

➢ **Software Engineering Approach – Phase Development Process, Project Management**

Software Engineering Approach

- The basic approach of software engineering is to separate the process for developing software from the developed product (i.e developed software).
- The idea is that the software process determines the quality of the product and productivity achieved. So we must have to focus on the software process.

- Design of proper software processes and their control then become a key goal of the software engineering. Software engineering focuses on the process for producing the products.
- So, there are two key goal phase (1) Phased Development Process (2) Managing the process.

1. **Phased Development Process:**

- A development process consists of various phases, each phase ending with a defined output.
- The phases are performed in an order specified by the process model being followed.
- The main reason for having a phased process is that it breaks the problem of developing software into successfully performing a set of phases, each handling a different concern of software development.
- A phased development process is central of the software engineering approach for solving the software crisis.

models proposed for developing software For small problems, these activities may not be done explicitly. The basic phases are as follow.

Requirement analysis:

- Requirements analysis is done in order to understand the problem the software system is to solve. For complex system, even determining what is needed is a difficult task.
- The goal of the requirements activity is to document the requirement in a software requirement specification document.

Software Design:

- The purpose of the design phase is to plan a solution of the problem specified by the requirement document.
- This phase is the first step in moving from the problem domain to the solution domain.

Coding:

- Once the design is complete, most of the major decisions about the system have been made.

- The goal of the coding phases is to translate the design of the system into code in a given programming language.

Testing:

- Testing is the major quality control measure used during software development.
- Its basic function is to detect defects in the software.
- After coding, computer programs are available that can be executed for testing purposes.

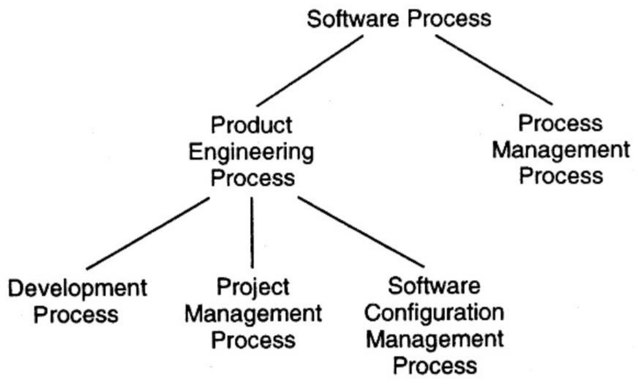
2. **Project (process) management:**
   - A phase development process is central to the software engineering approach.
   - However, a development process does not specify how to allocate resources to the different activities in the process.
   - There is not specify things like schedules for the activities, how to divide work within a phase, how to ensure that each phase is being done properly.
   - These issues relating to managing the development process of a project are handle though project management.
   - The management activity typically revolves around a plan.
   - A software plan forms the baseline that is heavily used for monitoring and controlling the development process of the project.
   - Managing a process require information upon which the management decision is based.
   - Otherwise, even the essential questions-is the schedule in a project is being met, what is the extent of cost overrun, are quality objectives being met – cannot be answer.
   - Software metrics are quantifiable measures.
   - There are two types of metrics use for software development: product metrics and process metrics.
   - Product metrics are used to quantify characteristics

of the product being developed, i.e, the software.
- Process metrics are used to quantify characteristics of the process being used to develop the software.
- Metrics and measurement are necessary expect of managing a software development project.

➢ Software Process & It's Characteristics

- A software process is a systematic, organized sequence of activities, methods, and practices that guide the development and maintenance of software products.
- Software processes in software engineering refer to the methods and techniques used to develop and maintain software.
- A software process is a set of activities, together with ordering among them, such that if the activities are performed properly and in accordance with the ordering, the desire result is produced.
- The process that deal with the technical and management issues of software development is called a software process.
- Many different types of activities need to be performed to develop software. All these activities together comprise the software processes.
- A successful project is the one that satisfies the expectations on all the three goals of cost, schedule and quality.
- A project's process specification defines the tasks the project should perform, and the order in which they should be done.
- A project's process may utilize some process model.

✓ Components of software processes


Software Process
Product Engineering Process
Process Management Process
Development Process
Project Management Process
Software Configuration Management Process

There are main two component of software process – **Product Engineering Process** and **Process Management Process.**

1. **Product Engineering Process:**
   - process focus on project and product. Its main objective is to produce desire product. It is consisting of three other processes.
   - **Development Process** : It is specifies the development and quality assurance activities that need to be performed.
   - **Project Management Process**: It is specifies that how to plan and control these activities so that cost, schedule, quality and other activities are met.
   - **Software Configuration Control Process**: It is main focus on evolution and correct version as project proceeds.
2. **Process Management Process:** The main objective is to improve capability of the process to produce quality goods (software) at low cost.

Characteristics of software

- **Functionality:** The primary ability of the software to perform its intended tasks and meet the user's needs.
  - It is one of the most important characteristics of software, as it determines the usefulness of the software for the intended purpose. Examples of functionality in software include:
  - Data storage and retrieval
  - Data processing and manipulation
  - User interface and navigation
  - Communication and networking
  - Security and access control

- **Reliability:** The ability of the software to perform its functions without failures under specified conditions over a period of time.

  Reliability is a characteristic of software that refers to its ability to perform its intended functions correctly and consistently over time. Reliability is an important aspect of software quality, as it helps ensure that the software will work correctly and not fail unexpectedly.

Examples of factors that can affect the reliability of software include:

1. Bugs and errors in the code
2. Lack of testing and validation
3. Poorly designed algorithms and data structures

- **Usability:** How easy and intuitive it is for users to understand, learn, and operate the software effectively.
  It refers to the extent to which the software can be used with ease. the amount of effort or time required to learn how to use the software.

- **Efficiency:** How well the software uses system resources like processor time and memory to perform its tasks.

  Efficiency is a characteristic of software that refers to its ability to use resources such as memory, processing power, and network bandwidth in an optimal way. High efficiency means that a software program can perform its intended functions quickly and with minimal use of resources, while low efficiency means that a software program may be slow or consume excessive resources.

  Examples of factors that can affect the efficiency of the software include:

  1. Poorly designed algorithms and data structures
  2. Inefficient use of memory and processing power
  3. High network latency or bandwidth usage

- **Maintainability:** The ease with which the software can be modified, corrected, or updated to meet evolving requirements or fix bugs.
  It refers to the ease with which modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

- **Portability:** The ability of the software to be transferred and operated on different hardware, operating systems, or environments.

  A set of attributes that bears on the ability of software to be transferred from one environment to another, without minimum changes.

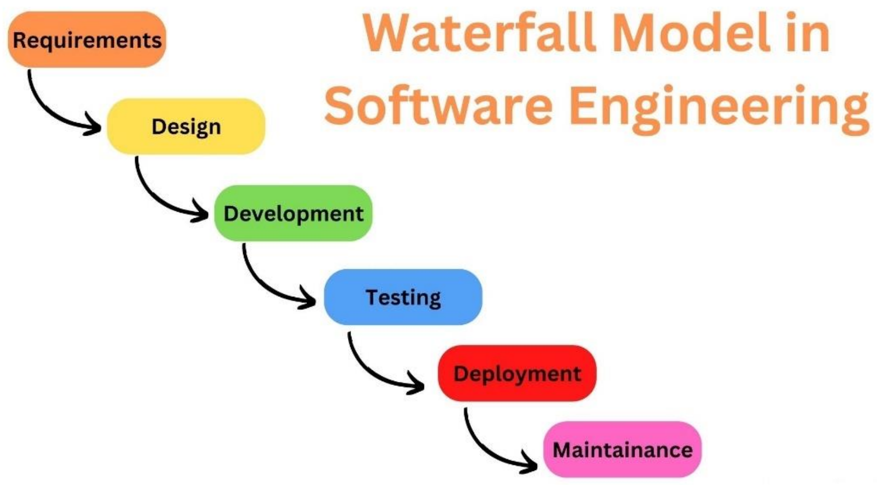
➢ **Software Development Process Models – Water Fall Model, Iterative Enhancement.**

1. Water Fall Model:

- The Waterfall Model is a Traditional Software Development Methodology.
- It is a linear and sequential approach to software development that consists of several phases.
- This classical waterfall model is simple and idealistic. It is important because most other Types of Software Development Life Cycle Models are a derivative of this.

Phases of Waterfall Model

Classical Waterfall Model divides the life cycle into a set of phases. The development process can be considered as a sequential flow in the waterfall. The different sequential phases of the classical waterfall model are follow:



1. Requirements Analysis and Specification

Requirement Analysis and specification phase aims to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.

Requirement Gathering and Analysis:

- Firstly, all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed.
- The goal of the analysis part is to remove incompleteness and inconsistencies

Requirement Specification:

- These analyzed requirements are documented in a software requirement specification (SRS) document.
- SRS document serves as a contract between the development team and customers.

2. Design

The goal of this Software Design Phase is to convert the requirements acquired in the SRS into a format that can be coded in a programming language.

It includes high-level and detailed design as well as the overall software architecture. A Software Design Document is used to document all of this effort (SDD).

High-Level Design (HLD): This phase focuses on outlining the broad structure of the system. It highlights the key components and how they interact with each other, giving a clear overview of the system's architecture.

Low-Level Design (LLD): Once the high-level design is in place, this phase zooms into the details. It breaks down each component into smaller parts and provides specifics about how each part will function, guiding the actual coding process.

3.Testing and Deployment

Testing:

- Integration of different modules is undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over several steps.
- During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested.
- Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this. **System testing consists of three different kinds of testing activities as described below.**

  - **Alpha testing**: Alpha testing is the system testing performed by the development team.
  - **Beta testing**: Beta testing is the system testing performed by a friendly set of customers.
  - **Acceptance testing**: After the software has been delivered, the customer performs acceptance testing to determine whether to accept the delivered software or reject it.

4.Deployment:

- Once the software has been thoroughly tested, it's time to deploy it to the customer or end-users.
- This means making the software ready and available for use, often by moving it to a live or staging environment.
- During this phase, we also focus on helping users get comfortable with the software by providing training, setting up necessary environments, and ensuring everything is running smoothly.
- The goal is to make sure the system works as expected in real-world conditions and that users can start using it without any hitches.

5. Maintenance

In Maintenance Phase is the most important phase of a software life cycle. The effort spent on maintenance is 60% of the total effort spent to develop a full software. There are three types of maintenance.

1. Corrective Maintenance:
2. Perfective Maintenance:
3. Adaptive Maintenance:

Advantages of waterfall model:

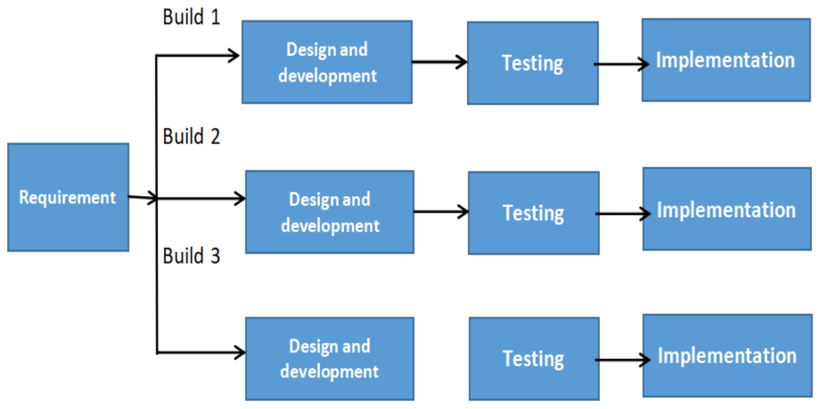
- Relatively simple to understand.
- Each phase of development proceeds sequentially.
- Allows managerial control where a schedule with deadlines is set for each stage of development.
- Helps in controlling schedules, budgets, and documentation.

Disadvantages of waterfall model:

- Requirements need to be specified before the development proceeds.
- The changes of requirements in later phases of the waterfall model cannot be done.
- No user involvement and working version of the software is available when the software is developed.

➢ Explain Iterative Model

- Software development uses a dynamic and adaptable method called the iterative enhancement Model.
- The iterative enhancement model encourages a software product's ongoing **evolution and improvement**.
- This methodology is noticeable due to its concentration on **adaptability, flexibility and change responsiveness**.
- It makes it easier for a product to evolve because it gives developers the freedom to progressively enhance the software, making sure that it complies with evolving specifications, user demands, and market demands. This helps products evolve more easily.



- The Iterative Enhancement Model creates an environment where development teams can more effectively adjust to changing requirements by segmenting the software development process into smaller, more manageable parts.
- Every iteration improves on the one before it, adding new features and fixing problems found in earlier stages.
- Members of the team, stakeholders and end users are encouraged to collaborate and communicate continuously to make sure the software meets changing needs and expectations.
- Until the software is finished being built, the iteration process is carried out, which involves giving the user the increments.

Advantages of Iterative model:

- Problems and risks can be identified and addressed early in the development process, reduces chances of issue for future stages.
- Feedback and constant client involvement are encouraged to make sure the finished product lives up to user expectations.
- Every iteration is put through testing and improvement, leading to higher quality product.

Disadvantages of iterative model:

- Higher cost
- Due to constant changes, there may be delays in documentation, making it more difficult to maintain comprehensive documentation.

- Continuous customer engagement may not be possible in all scenarios, which impacts the effectiveness of the model.

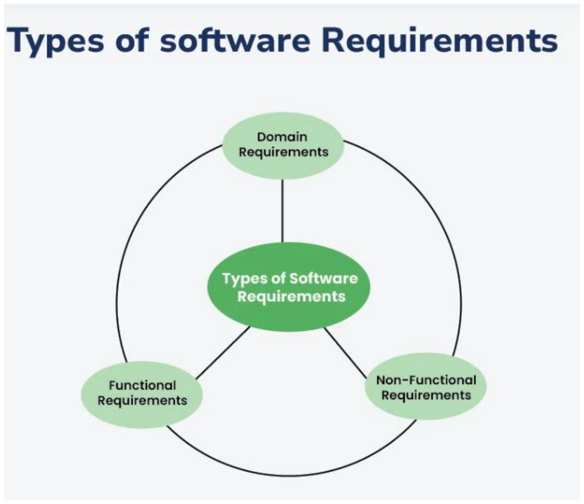
➢ Define Software Requirements

- Classification of Software Requirements is important in the software development process.
- It organizes our requirements into different categories that make them easier to manage, prioritize, and track.
- The main types of Software Requirements are functional, non-functional, and domain requirements.
- A condition or capability needed by a user to solve a problem or achieve an objective
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents

Types of Software Requirements

Software Requirements are mainly classified into three types:

1. Functional requirements
2. Non-functional requirements
3. Domain requirements


Types of software Requirements
Domain Requirements
Types of Software Requirements
Functional Requirements
Non-Functional Requirements

1. **Functional Requirements**

   Functional requirements describe what the software should do. They define the functions or features that the system must have.

2. **Non-functional Requirements**
   Non-functional requirements describe how the software performs a task rather than what it should do. They define the quality attributes, performance criteria, and constraints.

3. **Domain Requirements**
   Domain requirements are specific to the domain or industry in which the software operates. They include terminology, rules, and standards relevant to that particular domain.

➢ **Need For SRS, Role of SRS**
Need of SRS:

- The origin of most software systems is in the needs of some clients. The software system
- itself is created by some developers. Finally, the completed system will be used by the end users.
- A basic purpose of the SRS is to bridge this communication gap so they have a shared vision of the software being built.
- We have needs of software requirement specification as :
    - An SRS establishes the basis for agreement between the client and the supplier on what the software product will do.
    - An SRS provides a reference for validation of the final product.
    - A high-quality SRS is a prerequisite to high-quality software.
    - A high-quality SRS reduces the development cost.
    - Its need to clear any communication problems between the client and the developer.
    - It is need to minimize the amount of time and efforts of developers to achieve desired software goals.
    - Its need to traces out the functional and non-functional requirements of the product.
    - SRS is needs to describe the requirements of software in detail.
    - SRS is needs to decomposition of complex problem so that developer can easily develop software.
    - SRS needs to include a set of use cases that describe user interactions that the software must provide.
    - It is required to permits a rigorous assessment of requirements before design can begin and reduces later redesign.

- SRS required preventing software project failure.

Role of SRS

Software Requirement Specification (SRS) plays an important role in Software Engineering.

We can see it in the following way:

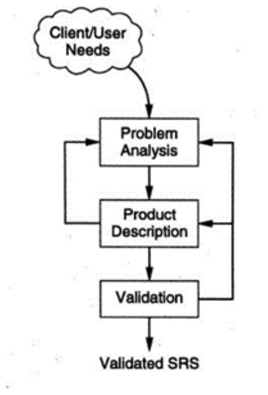
- It is play role as Congruency between the customers and the suppliers on what the software product is to do.
- It is play role to reduce the development effort.
- It is play role as base for estimating costs and schedules.
- It is play role as a baseline for validation and verification.
- It is play role as transformer. The Software Requirement Specification makes it easier to transfer the software product to new users or new machines.
- It is also play role as base for enhancement. Because the Software Requirement
- Specification discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product.
- It is play important role in design of inputs.
- It is play a role in providing opportunity for Review with end user.

- **Requirement Process -Problem Analysis, Requirement, Specifications, Validation :**

1. Requirement Process

- The requirement process is the sequence of activities that need to be performed in the requirements phase and that culminate in producing a high-quality document containing the SRS.
- The requirements process typically consists of three basic tasks: problem or requirement analysis, requirements specification, and requirements validation.
- Problem analysis often starts with a high-level “problem statement.”
- Requirement specification clearly specifying the requirements in a document likes representation, specification language, tools etc.
- Requirement validation focus on ensuring that what has been specified in the SRS are indeed

- all the requirements of the software and making sure that the SRS is of good quality.



Problem Analysis:
- The basic aim of problem analysis is to obtain a clear understanding of the needs of the clients and users.
- The analysts have to ensure that the real needs of the clients and the users are uncovered, even if they don't know them clearly.
- The analysts are not just collecting and organizing information about the client's organization and its processes.
- The basic principle used in analysis is the same as in any complex task – divide and conquer.
- These approaches and methods are given below:

Informal Approach:
- The informal approach to analysis is one where no defined methodology is used.
- In this approach, information about the system is obtained by interaction with the client, end users, questionnaires, study of existing documents etc.
- There is no formal model is built of the system.

Data Flow Modeling:
- Data-flow based modeling is also known as structured analysis technique.
- It is used function-based decomposition while modeling the problem. It focuses on the functions performed in the problem domain and the data consumed and produced by these functions.

- It is top-down refinement approach, which was originally called Structured Analysis and Specification.

Data Flow Diagram and Data Dictionary:

(1) Data Flow Diagram

- Data flow diagram are commonly used during problem analysis. Data flow diagrams are quite general and are not limited to problem analysis for software requirement specification.
- Data flow diagrams are very useful in understanding a system and can be effectively used during analysis.

(2) Data Dictionary:

- The data dictionary is a repository of various data flowed defined in a DFD.
- The associated data dictionary states precisely the structure of each data flow in the DFD.

(3) The structured Analysis Method:

- The basic view of this approach is that each system can be viewed as a transformation function operating within an environment that takes some inputs from the environment and produces some outputs for the environment.
- The overall transformation function is too complex to handle as a single function. So, the function should be partitioned into sub functions.

Object-Oriented Modeling:

- In object-oriented modeling, a system is viewed as a set of objects. The objects interact with each other through the services they provide.
- The goal of modeling is to identify the object that exist in the problem domain, define the classes by specifying what state information they encapsulate and what services they provide, and identify relationships that exist between objects of different classes, such that the overall model is such that it supports the desired user services.
- Here some concepts are needed to understand for discuss analysis.

Basic concepts and notation:

- Following are some concepts are useful to understand object oriented modeling technique.

  - Object : It is real world entity likes people, place, animal etc.
  - Class: It is a group of similar type of objects.
- Class diagram: It is represent a structure of the problem graphically using a precise notation.
- Generalization: This structure is used by a class to inherit some attributes and services of a general class.
- Specialization: This structure is used by a class that inherit some special
- attributes and services to make the specialized class from the general class.
- Aggregation: This structure is used to make an object from composing other objects.

Performing Analysis:

- There can be no algorithm to perform analysis or SRS.
- There are given a set of guideline, according to which the major steps in the analysis are given below.
- Identifying objects and classes
- Identifying Structure
- Identifying Attributes
- Identifying Association
- Defining services

Prototyping:

- Prototyping takes a different approach to problem analysis as compared to modeling-based approach.
- In prototyping, partial system is constructed, which is then used by the client, users and developers to gain a better understanding of the problem and the needs.
- A software prototype can be defined as a partial implementation of a system whose purpose is to learn something abut the problem being solved or the solution approach.

Requirement Specification:

- The understanding obtained by problem analysis forms the basis of requirements
- specification, in which the focus is on clearly specifying the requirements in a document.
- Issues such as representation, specification languages, and tools are addressed during this activity.
- As analysis produces large amounts of information and knowledge with possible redundancies, properly organizing and describing the requirements is an important goal of this activity.

Characteristics of an SRS

- To properly satisfy the basic goals, an SRS should have certain properties and should contain different types of requirements. Some of the desirable characteristics of an SRS are :

- **Correct**
  An SRS is correct if every requirement included in the SRS represents something required in the final system.

- **Complete**
  It is complete if everything the software is supposed to do and the responses of the software to all classes of input data are specified in the SRS.

- **Unambiguous**
  It is unambiguous if and only if every requirement stated has one and only one interpretation. Requirements are often written in natural language, which is inherently ambiguous. If the requirements are specified in a natural language, the SRS writer has to be especially careful to ensure that there are no ambiguities.

- **Verifiable**
  An SRS is verifiable if and only if every stated requirement is verifiable. A requirement is verifiable if there exists some cost-effective process that can check whether the final software meets that requirement.

- **Consistent**
  It is consistent if there is no requirement that conflicts with another.

- **Ranked for importance and/or stability**
  Together the performance and interface requirements and design constraints can be called nonfunctional requirements.

Components of an SRS

Completeness of specifications is difficult to achieve and even more difficult to verify. The basic issues an SRS must address are:

- **Functionality**
    - Functional requirements specify the expected behavior of the system—which outputs should be produced from the given inputs.
    - They describe the relationship between the input and output of the system.

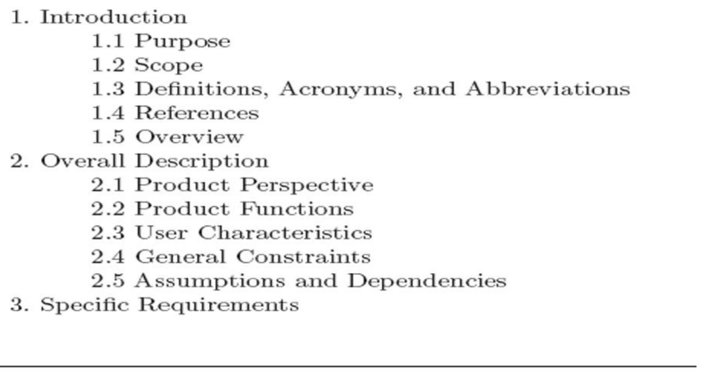
- **Performance**
    - The performance requirements part of an SRS, specifies the performance constraints on the software system.
    - There are two types of performance requirements: static and dynamic.

- **Design constraints imposed on an implementation**
    - There are a number of factors in the client's environment that may restrict the choices of a designer leading to design constraints.
    - Such factors include standards that must be followed, resource limits, operating environment, reliability and security requirements, and policies that may have an impact on the design of the system.

- **External interfaces**
    - In the external interface specification part, all the interactions of the software with people, hardware, and other software should be clearly specified.
    - For the user interface, the characteristics of each user interface of the software product should be specified.

Structure of a Requirements Document

- Requirements have to be specified using some specification language.
- Though formal notations exist for specifying specific properties of the system, natural languages are now most often used for specifying requirements.
- All the requirements for a system, stated using a formal notation or natural language,
- have to be included in a document that is clear and concise.
- IEEE guide to software requirements specifications.
- The general structure of an SRS is given in Figure.



1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions, Acronyms, and Abbreviations
    - 1.4 References
    - 1.5 Overview
2. Overall Description
    - 2.1 Product Perspective
    - 2.2 Product Functions
    - 2.3 User Characteristics
    - 2.4 General Constraints
    - 2.5 Assumptions and Dependencies
3. Specific Requirements

Figure 3.2: General structure of an SRS.

- The introduction section contains the purpose, scope, overview, etc., of the requirements document.
- The next section gives an overall perspective of the system - how it fits into the larger system, and an overview of all the requirements of this system.

Requirement Validation:

- It's a process of ensuring the specified requirements meet the customer needs.
- It's concerned with finding problems with the requirements.
- These problems can lead to extensive rework costs when these they are discovered in the later stages, or after the system is in service.
- During the requirements validation process, different types of checks should be carried out on the requirements. These checks include:
    - **Validity checks**: The functions proposed by user should be associated

with what the system needs to perform. You may find later that there are additional or different functions are required or not.

- **Consistency checks**: Requirements in the document shouldn't conflict or different description of the same function.
- **Completeness checks**: The document should include all the requirements and constrains.
- **Realism checks**: Ensure the requirements can actually be implemented using the knowledge of existing technology, the budget, schedule, etc.
- **Verifiability**: Requirements should be written so that they can be tested. This means you should be able to write a set of tests that demonstrate that the system meets the specified requirements.